Review Board User Manual

Release 2.0 beta 4 (dev)

Christian Hammond

February 25, 2014

CONTENTS

1 Getting Started **1**
- 1.1 Introduction . . . 1
- 1.2 What is Code Review? . . . 1
- 1.3 General Workflow . . . 2
- 1.4 Account Settings . . . 2

2 Dashboard **5**
- 2.1 Overview . . . 5
- 2.2 Navigation Sidebar . . . 5
- 2.3 Review Requests List . . . 6
- 2.4 Sorting . . . 6
- 2.5 Reordering Columns . . . 6
- 2.6 Customizable Columns . . . 7

3 Working with Review Requests **11**
- 3.1 Creating Review Requests . . . 11
- 3.2 Editing Fields . . . 14
- 3.3 Uploading Diffs . . . 16
- 3.4 Uploading File Attachments . . . 16
- 3.5 Publishing Review Requests . . . 17
- 3.6 Publishing Updates . . . 17
- 3.7 Closing Review Requests . . . 18

4 Reviewing **21**
- 4.1 Reviewing Diffs . . . 21
- 4.2 Reviewing File Attachments . . . 23
- 4.3 Reviewing Images . . . 24
- 4.4 Reviewing Markdown Files . . . 25
- 4.5 Issue Tracking . . . 26
- 4.6 Review Draft Banner . . . 28
- 4.7 Creating and Editing Reviews . . . 28
- 4.8 Publishing Reviews . . . 29
- 4.9 Approving Changes (Ship It!) . . . 29
- 4.10 Replying to Comments . . . 30

5 Searching **31**
- 5.1 Quick Search . . . 31
- 5.2 Full-Text Search . . . 31

6 Using Markdown **35**
- 6.1 Basic Markdown Syntax . . . 35

6.2 Code Samples 36
6.3 Escaping 37

CHAPTER

ONE

GETTING STARTED

1.1 Introduction

Review Board is a tool for reviewing source code, documentation and other text-based files. It offers a powerful web-based interface with broad browser support for managing review requests and reviewing code, as well as command line tools to simplify the review request submission process.

Review Board supports the following browsers:

- Firefox 3.0+
- Internet Explorer 9+
- Google Chrome

The following browsers are unsupported but known to work:

- Firefox 2.0
- Internet Explorer 8
- Opera 9+

Review Board is designed to fit in with existing *pre-commit review* and *post-commit review* methods.

1.2 What is Code Review?

1.2.1 Overview

Code review is the process of making pieces of source code available for other developers to review, with the intention of catching bugs and design errors before the code becomes part of the product.

Code review dramatically helps in the quality of products. By catching mistakes early, a lot of bad problems in a product can be avoided. Not all companies or developers take advantage of code review, but more and more are making it a part of their development culture and requirements.

1.2.2 Types of Code Review

There are two types of code review: pre-commit and post-commit.

Pre-commit review is a form of code review where code is reviewed *before* going into the codebase. In this method, a *diff file* is uploaded to Review Board, which reviewers can comment on, and once there's approval the code is committed to the repository.

Post-commit review is where the code is reviewed *after* going into the codebase. The code is committed to the repository and, at some point later, the code is reviewed. Any fixes that need to be made are then committed again later.

The advantage of pre-commit review over post-commit review is that mistakes are caught before they become part of the product. The downside is that development may take longer.

Review Board supports both styles of code review. For pre-commit reviews, an uncommitted change can be uploaded either through the web UI, or using the *rbt post* (http://reviewboard.org/docs/rbtools/dev/rbt/commands/post/#rbt-post) tool.

For post-commit reviews, depending on the repository type, Review Board allows you to browse the commit history and choose a change to review. All repository types are supported for post-commit review by using the *rbt post* (http://reviewboard.org/docs/rbtools/dev/rbt/commands/post/#rbt-post) tool to prepare a diff file for the change and upload it.

1.3 General Workflow

Review Board supports many different workflows, but most people use a *pre-commit review* model. The general process for using Review Board for pre-commit review is as follows:

1. Make a change to your local source tree.
2. *Create a review request* (page 11) for your new change.
3. Publish the review request and wait for your reviewers to see it.
4. Wait for feedback from the reviewers.
5. If reviewers have requested changes:
   (a) Update the code in your tree and generate a new diff.
   (b) Upload the new diff, specify the changes in the Change Description box, and publish.
   (c) Jump back to step 4.
6. If reviewers say "Ship It!"
   (a) Submit your change to the repository.
   (b) Click *Close* → *Submitted* on the review request action bar.

post-commit review and decentralized version control workflows may be different. If you've joined a company that uses Review Board, and you're unsure about your specific process, you'll want to talk to your employer to find out the specifics.

1.4 Account Settings

1.4.1 Overview

To get to the account settings page, go to the user menu at the top right of the page (signified by your gravatar image). From here, select *My Account*.

1.4.2 Groups

If Review Board has review groups, you can choose which groups you'd like to join using the checkboxes in this section. Review Requests which are assigned to groups that you are in will show up on your dashboard.

1.4.3 Settings

There are a few settings which allow you to affect how you interact with Review Board:

- **Time zone** This setting will change which time zone is used to show times and dates. This should be set to the time zone in your current location.
- **Enable syntax highlighting in the diff viewer** By default, code is shown using syntax highlighting. If you'd like to turn it off, de-select this item.
- **Always open an issue when comment box opens** By default, when you create a comment, the "Open an issue" checkbox will be pre-selected. If you prefer to opt-in to creating issues for each comment, de-select this item.

1.4.4 Authentication

The Authentication page allows you to change your password. This page may not be available if Review Board is configured to use an external authentication system like LDAP or Active Directory.

1.4.5 Profile

The Profile page allows you to change the real name and e-mail address associated with your account. These settings may not be available if Review Board is configured to use an external authentication system like LDAP or Active Directory. There is also a setting to control the privacy of your account:

- **Keep profile information private** Normally, the user profile page will show your real name, e-mail address, and when you last logged in. If you'd like to hide this information, select this item.

1.4.6 Gravatar Images

Review Board uses the gravatar system to associate photos or pictures with user accounts. To set your gravatar, go to http://gravatar.com/ and enter the email address used on your Review Board account.

CHAPTER

TWO

DASHBOARD

2.1 Overview

The dashboard is your primary method of accessing review requests. It displays detailed information on review requests at a glance, and allows filtering review requests.

After logging in to Review Board, you'll be taken to your dashboard. You can always get back to it by clicking *Dashboard* on the navigation banner near the top of the page.

2.2 Navigation Sidebar

The dashboard provides a navigation sidebar with the following items:

- *Outgoing*
  - *All*
  - *Open*
- *Incoming*
  - *Open*
  - *To Me*
  - *Starred*

It also lists each group you belong to, and each group you're watching. Each item also lists the number of review requests in that view. You can click on an item to be taken to that view of the dashboard.

2.2.1 Outgoing: All

This view shows every review request you have created, including those that are discarded or submitted. It works like the *Outgoing: Open* (page 5) but with your complete history.

2.2.2 Outgoing: Open

This view shows all review requests that you have filed that are open or are still drafts.

2.2.3 Incoming: Open

This is the default view. This view shows all review requests that have been either directly assigned to you or indirectly through a group you belong to.

This can be filtered down by selecting *To Me* or one of the group names under *Incoming Reviews*.

2.2.4 Incoming: To Me

This view shows all review requests that have been directly assigned to you.

2.2.5 Incoming: Starred

This view shows every review request that you have starred. This is useful for keeping track of review requests you are interested in that were not directly assigned to you.

2.3 Review Requests List

The main area of the dashboard lists the review requests belonging to that particular view. This is a detailed, sortable, customizable list.

Clicking on any review request in the list will take you to that particular review request, while clicking on a submitter's name will take you to the list of review requests submitted by that user.

2.4 Sorting

The review request list can be sorted by clicking on a column header. Clicking once will sort the column in ascending order, and clicking a second time will sort that column in descending order. The column will have a little up or down arrow indicating the sorting order. You can click the *X* next to clear sorting for that column.

The dashboard provides two-level sorting. You can primarilty sort by one column but in the case of multiple entries for a particular submitter, timestamp, etc., you can have secondary sorting on another column. This is set by simply clicking one column (which will be the secondary column) and then clicking another column (which will be the primary).

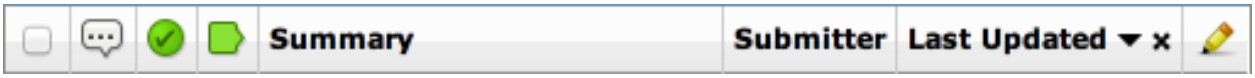
The primary column is indicated by a black up/down arrow, and the secondary column is indicated by a lighter grey up/down arrow.

Sorting options are saved across sessions.

2.5 Reordering Columns

Columns in the dashboard can be reordered by clicking and dragging the column. The columns will reorder as you're dragging to show you the new layout, and when you release the mouse cursor the order will be saved.

2.6 Customizable Columns



Different users have different things they want to see in the dashboard. You can change which columns are shown and which aren't by clicking the pencil icon to the right of the columns. A pop-up menu will appear showing which columns are shown and which aren't.

The following are available columns you can choose from:

2.6.1 Branch

Shows the branch information listed on the review request.

2.6.2 Bugs

Shows the bug IDs listed on the review request.

2.6.3 Diff Size

Shows a count of the removed and added lines of code in the latest revision of the diff.

2.6.4 Diff Updated

Shows the timestamp of the last diff update. This is color-coded to indicate the age.

2.6.5 Diff Updated (Relative)

Shows the timestamp of the last diff update, relative to now. This is color-coded to indicate the age.

2.6.6 Last Updated

Shows the timestamp of the last update to the review request. This is color-coded to indicate the age.

2.6.7 Last Updated (Relative)

Shows the timestamp of the last update to the review request, relative to now. This is color-coded to indicate the age.

2.6.8 My Comments

Shows a green comment flag if you have any unpublished comments on the review request, or a blue comment flag if you have published comments. This allows you to quickly see which review requests you've addressed.

2.6.9 New Updates

Shows a message bubble icon for any review requests that have been updated or have had discussion since you've last seen it. This does not apply to review requests you haven't yet looked at.

2.6.10 Number of Reviews

Shows how many reviews have been made on the review request.

2.6.11 Posted Time

Shows the timestamp when the review request was first posted. This is color-coded to indicate the age.

2.6.12 Posted Time (Relative)

Shows the timestamp when the review request was first posted, relative to now. This is color-coded to indicate the age.

2.6.13 Repository

Shows the repository that the review request is against.

2.6.14 Review Request ID

Shows the ID number of the review request.

2.6.15 Select Rows

Shows a checkbox that allows you to select the row. When one or more review requests are selected, the sidebar will contain commands to close the selected review requests.

2.6.16 Ship It!

If there are open issues, this shows a count of the open issues in a yellow bubble. If there are no open issues, this will show a count of reviews where someone has marked "Ship It!"

2.6.17 Starred

Shows a star indicator that can be toggled. When toggled on, the review request is starred, meaning you'll be CC'd on any discussion. Toggling it off will remove you from the CC list.

2.6.18 Submitter

Shows the username of the submitter.

2.6.19 Summary

Shows the summary text of the review request.

2.6.20 Target Groups

Shows a list of the assigned groups for each review request.

2.6.21 Target People

Shows a list of the assigned people for each review request.

2.6.22 To Me

Shows a chevron for review requests which directly list you in the "people" field.

CHAPTER

THREE

WORKING WITH REVIEW REQUESTS

3.1 Creating Review Requests

A review request consists of, at a minimum, a summary, a description, and some reviewers. A review request usually has a diff, but it can be useful to have review requests that only have file attachments.

There are two main ways of submitting a new review request: through the web UI and through *rbt post* (http://reviewboard.org/docs/rbtools/dev/rbt/commands/post/#rbt-post). We recommend the latter when posting diffs for review, as it does a lot of work for you, and is actually required for some version control systems, such as Perforce.

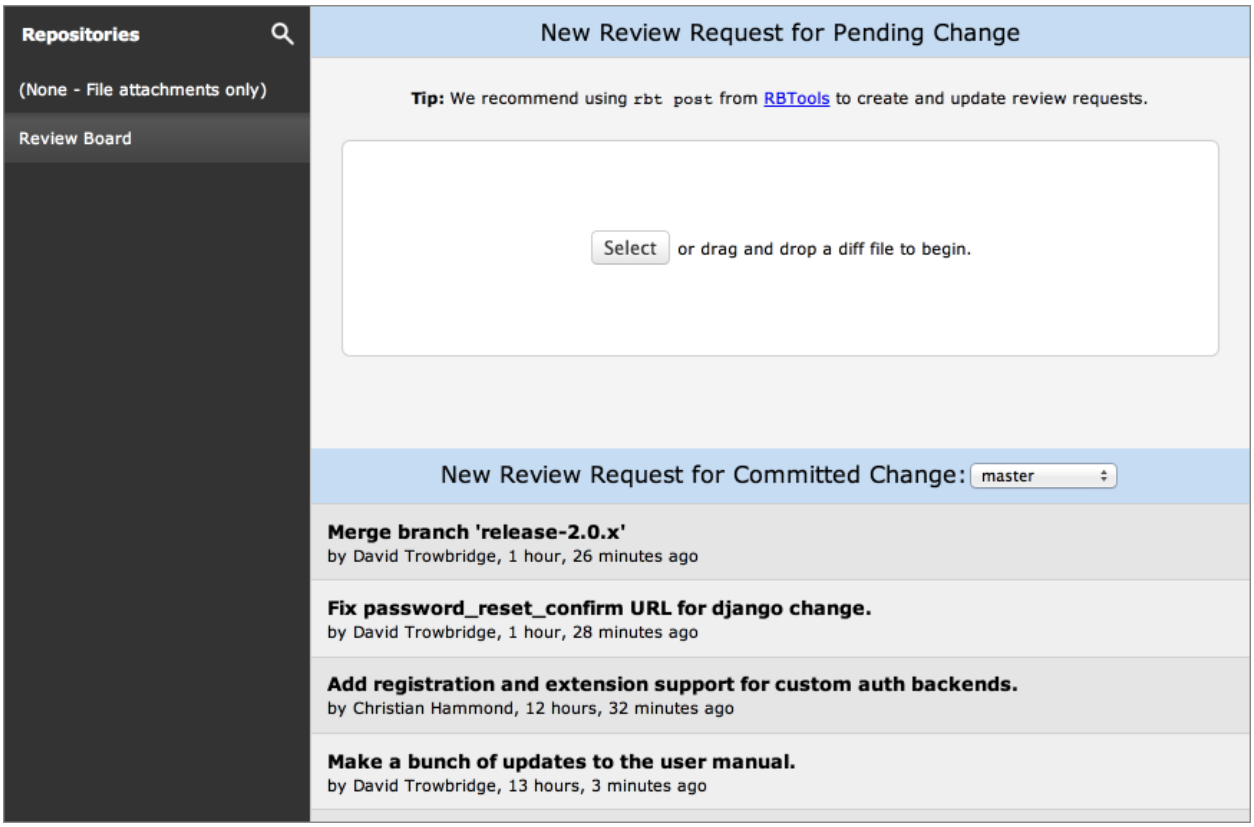
3.1.1 Using `rbt post` to Create Review Requests

RBTools (http://reviewboard.org/docs/rbtools/dev/#rbt) is a set of command-line tools that can be installed on each client system. There are many tools in the RBTools package, but the most important one is *rbt post* (http://reviewboard.org/docs/rbtools/dev/rbt/commands/post/#rbt-post), which will look into your development directory and post changes to Review Board.

Using RBTools is the recommended way of posting changes, and for some version control systems or hosting services, may be the only supported way (due to the built-in diff tools generating files with insufficient information). For more information, see the *RBTools documentation* (http://reviewboard.org/docs/rbtools/dev/#rbt).

3.1.2 Using the Web UI to Create Review Requests

To post a review request through the web UI, click *New Review Request* in the upper-left of a page. This will take you to a page where you can create your review request.


Repositories
(None - File attachments only)
Review Board
New Review Request for Pending Change
Tip: We recommend using rbt post from RBTools to create and update review requests.
Select or drag and drop a diff file to begin.
New Review Request for Committed Change: master
Merge branch 'release-2.0.x'
by David Trowbridge, 1 hour, 26 minutes ago
Fix password_reset_confirm URL for django change.
by David Trowbridge, 1 hour, 28 minutes ago
Add registration and extension support for custom auth backends.
by Christian Hammond, 12 hours, 32 minutes ago
Make a bunch of updates to the user manual.
by David Trowbridge, 13 hours, 3 minutes ago

On this page, you are presented with a variety of options for creating a new review request. On the left-hand side is a list of all the repositories configured on the server.

The first step in creating a new review request is to select one of these. Because some large Review Board servers may have dozens of configured repositories, you can search for the one you want by clicking the magnifying glass.

Pre-commit Review Requests

When doing *pre-commit review*, your review request will represent a pending change which is not committed to a repository. To upload the change to Review Board, you'll need to create a diff file.

The diff you provide needs to be in *unified diff* format, and must have revision information embedded in the file. CVS, Subversion, Git, Mercurial and Bazaar provide this information. If you use Perforce, though, you'll need to use *rbt post* (http://reviewboard.org/docs/rbtools/dev/rbt/commands/post/#rbt-post) instead of the web UI. Note that this means you can't just use a standard diff generated by the **diff** tool. You'll need to consult your version control system's documentation for information on generating diffs.

Once you've created your diff, select a repository. You can then either drag-and-drop your diff file onto the top section under *New Review Request for Pending Change* or browse for it using the *Select* button.

In most cases, once you've selected your diff file, the review request will be created. For some types of repositories, you may be prompted for additional information.

Perforce repositories:

- *Change Number*:

  The changeset number representing the change the diff is generated from.

Subversion repositories:

- *Base Directory*:

  The relative path of the directory you were in when you generated the diff, based on the repository. For example, if you locally have a checkout of `trunk/reviewboard` named `reviewboard`, and that was the directory you were in when you made the diff, then the base diff path would be `/trunk/reviewboard`.

  This may also depend on how Review Board was configured. Consult your administrator if you have problems.

Post-commit Review Requests

New in version 2.0.

The web UI now allows you to create review requests for *post-commit review*. This feature is not available for all types of repositories.

Once you select a supported repository, the bottom half of the "New Review Request" page will have a table labeled *New Review Request for Committed Change*. By default, commits are shown for a default branch (like "trunk" in Subversion or "master" in Git). In the header is a drop-down box which allows you to switch branches.

Below the header is a list of commits on the selected branch, starting with the most recent. As you scroll down, that list will update automatically with older changes. To create a post-commit review request, just click on one of the commits. Review Board will download the diff and change description and prepare a new review request. From there, just assign reviewers and publish.

If a committed change already has a review request, this will be denoted with an icon. In this case, clicking on the commit will jump to the existing review request rather than creating a new one.

Review Requests for File Attachments Only

If you're intending to post a review request only for reviewing file attachments, select *(None - File attachments only)* on the left-hand side. You'll then be presented with a button to create a blank review request. You will not be able to add any diffs to this review request, but after creating it, you will be able to add attachments.

No other fields will need to be provided when using this option.

3.1.3 Finishing the Review Request

Once you've posted your initial diff through the web UI or *rbt post* (http://reviewboard.org/docs/rbtools/dev/rbt/commands/post/#rbt-post), you'll have a draft of your review request posted. Nobody but you will be able to see the review request until you publish it.

Before you publish the review request, you'll need to fill out the summary, description and reviewers. Some of these fields may be filled in depending on your repository type, the tool you used to post the review request, and any defaults your administrator has set up for the reviewers (see *managing-default-reviewers* if you're the administrator).

File attachments can be added with the *Update* → *Add File* menu item or by dragging and dropping files onto the page in the browser.

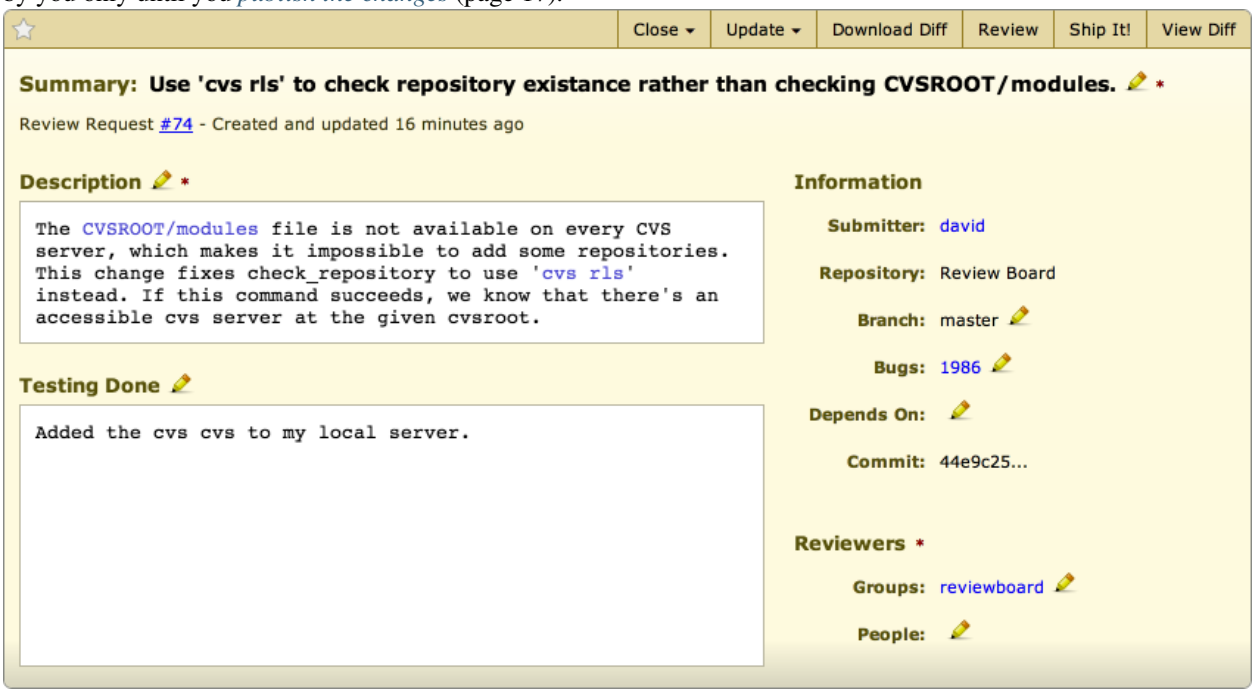
You can preview your diff by clicking *View Diff* in the review request action bar (in the top-right of the review request). Uploading a new diff (either using the *Update* → *Update Diff* menu item or with *rbt post* (http://reviewboard.org/docs/rbtools/dev/rbt/commands/post/#rbt-post)) before publishing will replace the diff.

When you're finished, click *Publish Review Request* in the green draft banner above the review request. You can also choose to discard the review request by clicking *Discard*.

Depending on your particular setup, publishing a review request may send an e-mail out to the reviewers, letting them know your change is ready to review.

3.2 Editing Fields

Updating review requests is very much like *creating review requests* (page 11). Once you have a review request out there, you can upload a new diff, add new file attachments, or modify the fields. Any changes you make will be seen by you only until you *publish the changes* (page 17).


Close
Update
Download Diff
Review
Ship It!
View Diff
Summary: Use 'cvs rls' to check repository existance rather than checking CVSROOT/modules.
Review Request #74 - Created and updated 16 minutes ago
Description
The CVSROOT/modules file is not available on every CVS server, which makes it impossible to add some repositories. This change fixes check_repository to use 'cvs rls' instead. If this command succeeds, we know that there's an accessible cvs server at the given cvsroot.
Testing Done
Added the cvs cvs to my local server.
Information
Submitter: david
Repository: Review Board
Branch: master
Bugs: 1986
Depends On:
Commit: 44e9c25...
Reviewers
Groups: reviewboard
People:

3.2.1 Changing Fields

Most fields on a review request can be changed, with the exception of *Submitter*, *Commit* and *Repository*.

To change a field, either click on the field (in the case of *Description* and *Testing Done*) or click on the pencil icon. A text box will appear allowing you to modify the value.

To save a field, press the `Enter` key or click *OK*. To revert your changes, press the `Escape` key or click :guilabel:‘Cancel’.

Summary

The *Summary* field is a short, one-line description of the change. It’s what people will see in their dashboard and in e-mail subject headers. You should aim to keep this short and as descriptive as possible.

Description

The *Description* field describes the change that will be reviewed. This is intended to provide enough information for reviewers to know what the change is about before they go to review it.

This field supports rich text using the *Markdown* language. See *Using Markdown* (page 35) for more information.

Testing Done

The *Testing Done* field describes how this change has been tested. This should cover any and all testing scenarios that have been done, in order to help reviewers feel more confident about the stability and design of the change.

This field supports rich text using the *Markdown* language. See *Using Markdown* (page 35) for more information.

Branch

The *Branch* field describes which branch your change applies to. This is a very free-form field and can contain any text.

Some examples may be:

- `trunk`
- `master`
- `my-feature`
- `release-2.0`
- `hotfix-branch -> release-2.0 -> main`

In the latter case, this could be used to show the series of branches that the change would be merged down to, starting at the branch where the change originated.

Bugs

The *Bugs* field is a comma-separated list of bug IDs that the change addresses. If the repository is configured with a bug tracker, the bug IDs will link to the reports on the bug tracker.

Depends On

The *Depends On* field is a comma-separated list of review request IDs which are used to indicate dependencies between changes. The IDs will link to the other review requests, allowing reviewers to take that information into account when reading the changes.

Groups

The *Groups* field is a comma-separated list of all review groups that should review the change.

When entering a group, Review Board will attempt to auto-complete the group. It will match against either the group's ID, or the group's name. While auto-completing, a drop-down of possible groups will be displayed, showing both the ID and name.

Review Board doesn't enforce that the groups must review the change before it can be submitted. This is a policy that is left up to each organization.

People

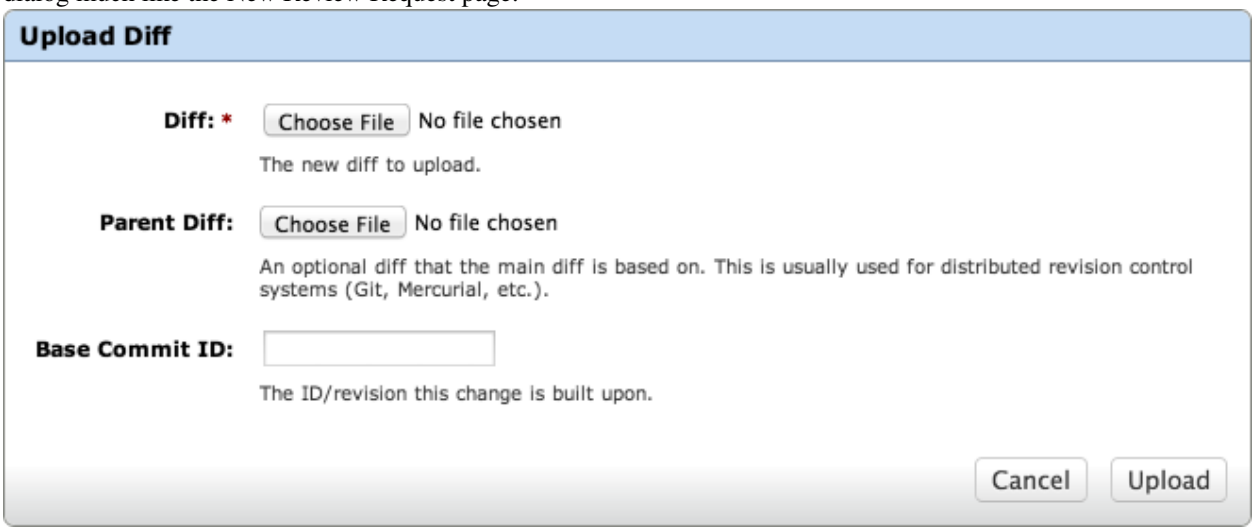
The *People* field is a comma-separated list of all the people that should review the change.

When entering a person, Review Board will attempt to auto-complete the person's information. It will match against either the person's username, or the person's first or last name. While auto-completing, a drop-down of possible people will be displayed, showing both the username and full name.

Review Board doesn't enforce that the people listed must review the change before it can be submitted. This is a policy that is left up to each organization.

3.3 Uploading Diffs

A review request's diff can be updated by clicking *Upload Diff* or *Update Diff* on the action bar. It will present a dialog much like the New Review Request page.


Upload Diff
Diff: *
Choose File
No file chosen
The new diff to upload.
Parent Diff:
Choose File
No file chosen
An optional diff that the main diff is based on. This is usually used for distributed revision control systems (Git, Mercurial, etc.).
Base Commit ID:
The ID/revision this change is built upon.
Cancel
Upload

See *Pre-commit Review Requests* (page 12) for more information on the fields.

Note: When updating the diff on a review request, you need to upload a full diff between the upstream code (what's present in the repository) and your change. You can't just upload a diff of the changes since the previous diff.

Note: It's usually best to use *rbt post* (http://reviewboard.org/docs/rbtools/dev/rbt/commands/post/#rbt-post) or another tool to update a diff, instead of uploading using this dialog. This will help guarantee a valid diff, and works around problems in some diff formats.

3.4 Uploading File Attachments

New in version 1.6.

Any type of file can be added to a review request. Log files, binaries, test data, audio, or anything else. This is especially helpful for files that are part of a change but can't be represented in diffs, as well as things like attaching a screenshot of changes that affect user interfaces.

Many types of files, including text and images, can be displayed on Review Board and reviewed by clicking on the areas where you want to add comments. This works just like the review process for source code. Simply click the thumbnail or click *Review* to begin reviewing the file.

Not all types of files have this support. For those files, clicking *Add Comment* will let you leave a comment that applies to the whole file.

See *Reviewing File Attachments* (page 23) for more information.

There are two ways to upload a file attachment:

1. Click *Update →Add File* on the action bar, browse for the file to upload, and optionally enter a caption for the file.
2. Drag the file from your file manager into the review request.

   When dragging over the browser window, a black overlay will appear, showing you where to drop the file to upload it.

   When you drop the file, the overlay will disappear and the file will upload.

   You can drag and upload multiple files at a time.

   Note that this requires a modern HTML5-capable browser, such as Google Chrome or Firefox 3.6+.

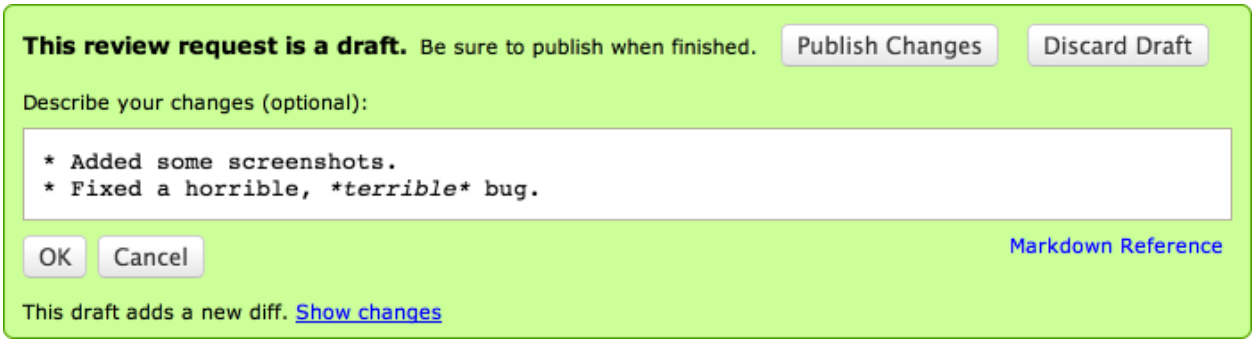
3.5 Publishing Review Requests

When you first create a review request, it will not be visible to anyone until it is published. Click *Publish* in the green bar at the top to make the review request public.

Depending on your particular setup, publishing a review request may send an e-mail out to the reviewers, letting them know your change is ready to be reviewed.

3.6 Publishing Updates

As soon as you make any change to a review request, a draft banner will appear above the review request asking you to optionally describe your changes. This is handy when you've actually uploaded a new diff and want to say what changed in that diff.


This review request is a draft. Be sure to publish when finished.
Publish Changes
Discard Draft
Describe your changes (optional):
* Added some screenshots.
* Fixed a horrible, *terrible* bug.
OK
Cancel
Markdown Reference
This draft adds a new diff. Show changes

When you're finished, click *Publish Changes*. If you decided you didn't want to make those changes, you can click *Discard Draft* instead.

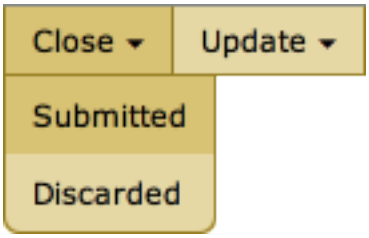
Depending on your particular setup, publishing a draft may send an e-mail out to the reviewers, letting them know your change is ready to be reviewed again.

The *Describe your changes* field supports rich text using the *Markdown* language. See *Using Markdown* (page 35) for more information.

3.7 Closing Review Requests

When a review request has been thoroughly reviewed, or the task has been abandoned, the review request can be closed.

There are two ways to close the review request: Closing as "submitted" or as "discarded." Both options are available under the *Close* menu on the review request's action bar.



A submitted review request is one where the change has been committed to the repository and no longer needs further reviews.

A discarded review request is one that has been abandoned and will no longer be updated or reviewed.

3.7.1 Close Descriptions

Once you've closed a review request, you can enter a more thorough description describing the commit or the reason you closed the review request.

For example, if you submitted the review request, you might specify the revision it was committed as. If you discarded, you might explain why.

The gray submitted or discarded banners have a field for the description. You can enter the text here and click *OK*. The description will be saved immediately, and other users will be able to see it.

This field supports rich text using the *Markdown* language. See *Using Markdown* (page 35) for more information.

This change has been marked as submitted.
Reopen for Review
Describe the submission (optional):
Pushed to master.

CHAPTER

FOUR

REVIEWING

4.1 Reviewing Diffs

4.1.1 Overview

Diffs can be reviewed in the Review Board diff viewer by clicking *View Diff* on the review request action bar.

The diff viewer provides a side-by-side view of the old and new versions of each modified file, complete with color coding and syntax highlighting. Lines can be commented on directly, allowing the developer to see exactly what part of code you're talking about in a review.

The top of the diff viewer shows the review request box, for reference. Inside that box toward the bottom is an indicator showing the current diff revision, sometimes followed by a listing of other revisions (if more than one diff was uploaded to this review request), then a file listing, and then the diffs.

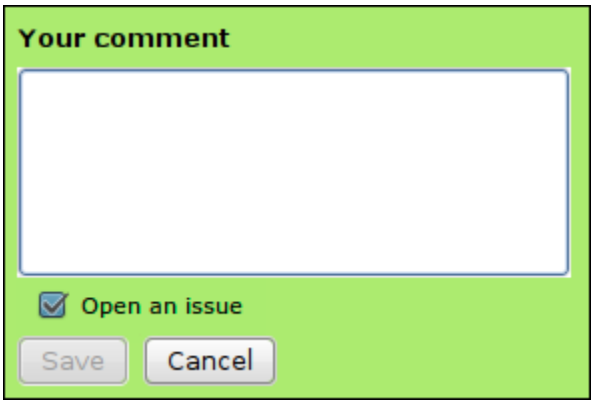
Depending on the number of files modified, the diff viewer may be split across multiple pages. You can jump to the pages using the paginator above or below the diffs.

4.1.2 Commenting on Lines

To comment on a line on a diff, simply click the line number. A comment dialog will appear giving you a text entry for writing your comment. When you're done, you can click *Save* to save the comment.

Furthermore you can assign a comment to multiple code lines. This option is especially useful to provide additional code context to discussions as all commented code will appear on the review request page.

To create a multiple line comment click and drag on the beginning line number down the column until you've selected all the lines needed for your comment.



Comments support rich text using the *Markdown* language. See *Using Markdown* (page 35) for more information.

The diff comment dialog supports issue tracking. See the section on *Issue Tracking* (page 26) for more information.

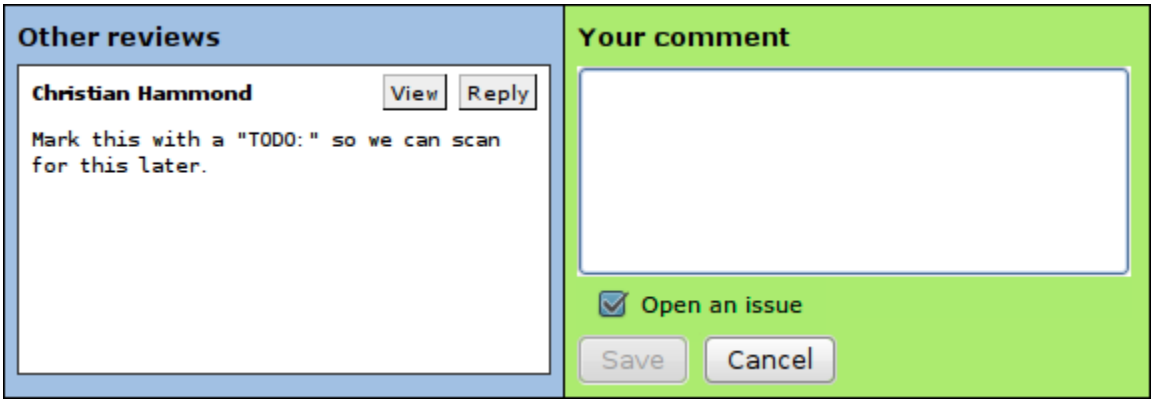
After saving a comment, a green comment flag will appear next to the first line in your selection, indicating that you have an unpublished comment. Click the line number or comment flag to pop open the comment box for your existing comment again.

4.1.3 Reading Existing Comments

The diff viewer will show blue comment flags along the left-hand side next to the line numbers that were already reviewed. The number inside the comment flag indicates how many comments were made on that line.

If you move the mouse cursor over the comment flag, a tooltip will appear showing a summary of the comments made.

If you click on the comment flag or the line number, the comment dialog will appear, along with a blue side panel on the left showing those existing comments. You can still write new comments in the green area of the comment box.

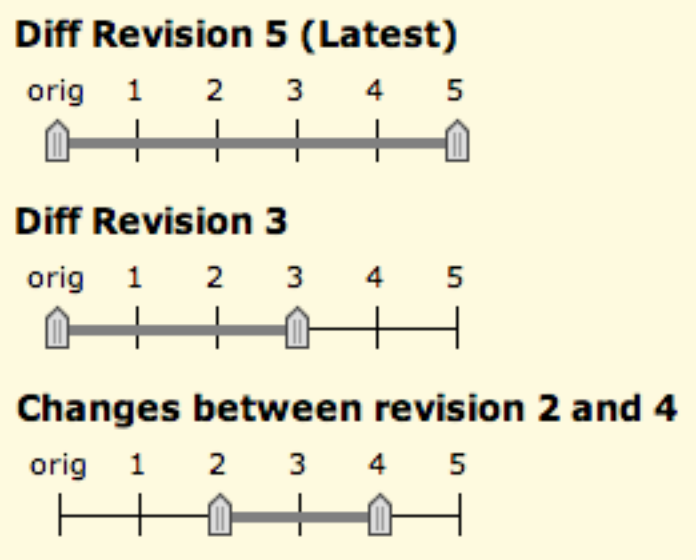


It's important to note that this is meant to be used as a reference to see if other people have already said what you plan to say. The comment box is **not** the place to reply to those comments. Instead, you can click the *Reply* link next to the particular comment, which will take you back to the review request page and open a reply box.

4.1.4 Viewing Other Diff Revisions

Every public revision of a diff that was posted is available for review. The diff revision selector allows you to look at previous versions of the change. To browse old revisions, set the handle on the left to *orig* (meaning the original version of the file) and the handle on the right to the revision you want to look at.

This is sometimes useful when you're in the middle of a review of a particular diff, go away for a bit, and then come back to discover that a new diff revision has been uploaded. In this case, a warning box will appear telling you that you have additional comments on an older revision, with a helpful link to jump back to that revision.



The diff viewer also allows you to do comparisons between diff revisions (called *interdiffs*). In other words, it lets you see what changes the developer has made since the previous version of the diff you looked at, which is very useful for large changes which require several iterations of review.

To view an interdiff between two diff revisions, set the two handles to the revisions you want to compare.

Note: Due to the way that the diff viewer works, if a newer diff is based on a newer revision of a file, you may see other changes made to that file between those revisions that has nothing to do with the review itself.

If you're a developer posting code and you want to sync your source tree, it's best to try to keep as many revisions of your change based on the same revision of the source tree as possible, in order to minimize the impact of this on your reviewers.

4.1.5 Keyboard Shortcuts

There are many keyboard shortcuts for navigating around the diff viewer:

- Previous file: `a`, `A`, `K`, `P`:, <, or `m`
- Next file: `f`, `F`, `J`, `N`, >
- Previous change: `s`, `S`, `k`, `p`, `,`
- Next change: `d`, `D`, `j`, `n`, `.`
- Previous comment: `[`, `x`
- Next comment: `]`, `c`
- Add comment to selected block: `r`, `R`

4.2 Reviewing File Attachments

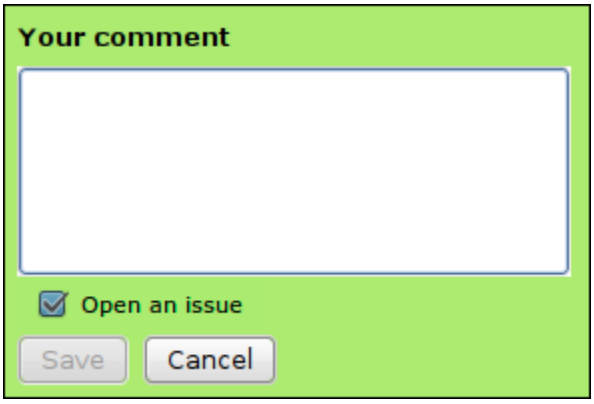
New in version 1.6.

File attachments can be reviewed using Review Board. Some file types, such as images, have special interfaces for doing reviews.

To learn how to review image files, read ref:*Reviewing Images <reviewing-images>*.

For other files, reviewers can download the files, and add comments to the attachment as a whole.

The file attachment box provides a *Add Comment* button. Clicking this button pops up a comment dialog that works exactly like the dialog you get when *reviewing diffs* (page 21).



Comments support rich text using the *Markdown* language. See *Using Markdown* (page 35) for more information.

The file attachment comment dialog supports issue tracking. See the section on *Issue Tracking* (page 26) for more information.

As of Review Board 1.6.0, there's no visual indication of an existing comment on the file attachment. However, you can see your comment and edit it by clicking *Add Comment* again.

4.3 Reviewing Images

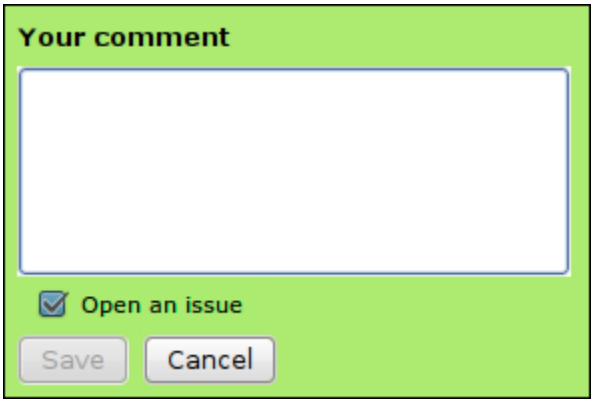
4.3.1 Overview

Much like diffs, portions of images can be reviewed. This is useful when presenting screenshots of a new dialog, for example, where the design and layout of the dialog is as important as the code constructing it.

To begin reviewing an image file, just click the thumbnail for the file attachment on the review request page.

4.3.2 Commenting on Regions

To place a new comment on part of the image, simply click and drag to select the desired area. This will pop open a new comment dialog, just like in the diff viewer.



Comments support rich text using the *Markdown* language. See *Using Markdown* (page 35) for more information.

The file attachment comment dialog supports issue tracking. See the section on *Issue Tracking* (page 26) for more information.

Once you're done writing your comment in the text area, click *Save* to save the comment.

A rectangle with a green number box will appear in the region you selected to indicate that you have a comment in that area. Click the box to pop open the comment box for your existing comment again.

4.3.3 Reading Existing Comments

When your mouse cursor is over the image, regions representing other comments may appear with blue number boxes in the corner. Each of these is a comment someone has made on another review.

If you move the mouse cursor over the region, a tooltip will appear showing a summary of the comments made.

If you click on the region, the comment dialog will appear, along with a blue side panel on the left showing those existing comments. You can still write new comments in the green area of the comment box.

It's important to note that this is meant to be used as a reference to see if other people have already said what you plan to say. The comment box is **not** the place to reply to those comments. Instead, you can click the *Reply* link next to the particular comment, which will take you back to the review request page and open a reply box.

4.4 Reviewing Markdown Files

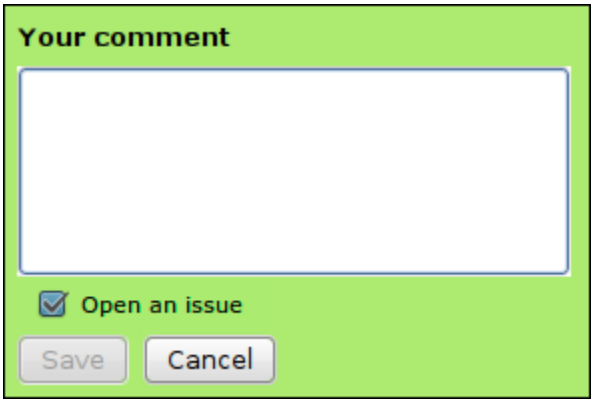
4.4.1 Overview

When writing documentation using the *Markdown* format, we often care much less about the source file than the resulting rich text document. Attaching a Markdown file (.md) to a review request will enable a special review UI that will render the document.

To begin reviewing a Markdown file, click the thumbnail or the "Review" link for the file attachment on the review request page.

4.4.2 Commenting on Blocks

As you hover your mouse over parts of the document, top-level blocks will be highlighted with a grey background. Clicking will pop open a new comment dialog, just like in the diff viewer.



Comments support rich text using the *Markdown* language. See *Using Markdown* (page 35) for more information.

The file attachment comment dialog supports issue tracking. See the section on *Issue Tracking* (page 26) for more information.

Once you're done writing your comment in the text area, click *Save* to save the comment.

After saving a comment, a green comment flag will appear next to the block you selected, indicating that you have an unpublished comment. If you want to edit your comment, click the block or comment flag to pop open the comment box again.

4.4.3 Reading Existing Comments

The Markdown review UI will show blue comment flags along the left-hand side of the document. The number inside the comment flag indicates how many comments were made on that block.

If you move the mouse cursor over the comment flag, a tooltip will appear showing a summary of the comments made.

If you click on the block or the comment flag, the comment dialog will appear, along with a blue side panel on the left showing those existing comments. You can still write new comments in the green area of the comment box.

It's important to note that this is meant to be used as a reference to see if other people have already said what you plan to say. The comment box is **not** the place to reply to those comments. Instead, you can click the *Reply* link next to the particular comment, which will take you back to the review request page and open a reply box.

4.5 Issue Tracking

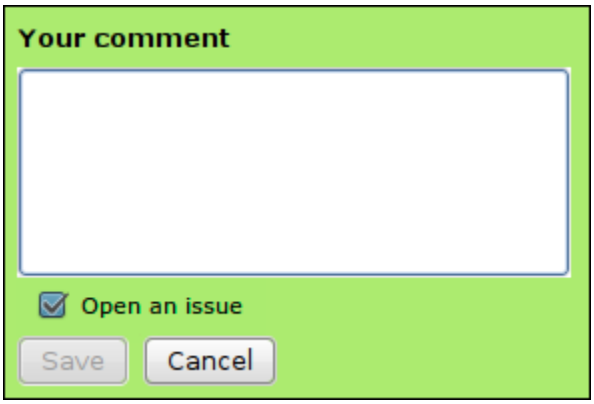
New in version 1.6.

When reviewing code or other files, some comments are more critical than others. The reviewer may just have a question or suggest something optional, but they may also have a critical issue that must be resolved before the change can be submitted.

Issue tracking enables reviewers to specify that their comment refers to a defect in the code or file that must be resolved. The owner of the review request will be able to see that issues were filed, and can resolve or discard them one-by-one. This is a faster alternative to replying to each comment with "Fixed."

4.5.1 Opening Issues

Comment dialogs contain an *Open an issue* checkbox. Comments with this checkbox checked will be filed as an open issue.



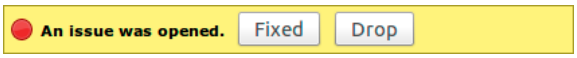
This checkbox is available when *reviewing diffs* (page 21), *images* (page 24), and other kinds of *file attachments* (page 23).

By default, *Open an issue* is checked for new comments.

The checkbox can also be toggled by pressing `Alt-I`.

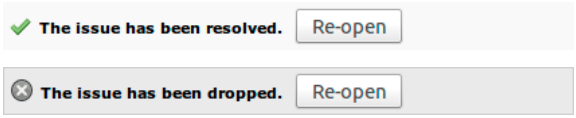
4.5.2 Responding to Issues

The owner of the review request will see extra buttons on the issue banners below the comment in the reviews for any issues that are opened. These buttons allow for quickly marking issues as either resolved or discarded.



Clicking *Fixed* will mark that particular issue as fixed, letting other reviewers see that you've taken care of the issue.

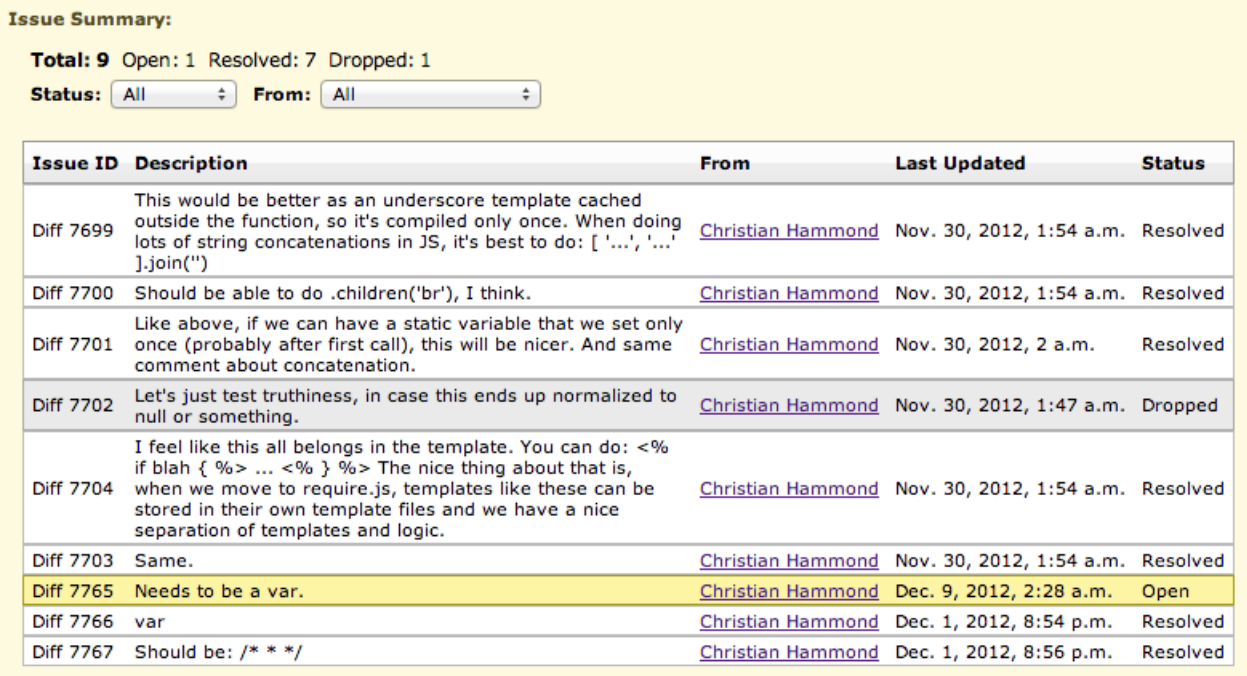
Clicking *Drop* will drop that issue. This signals to the reviewers that either their comment didn't make sense for one reason or another, or that there's a disagreement about the issue. Generally, this should be followed up with a comment.



Issues that are unintentionally closed one way or another can be re-opened by clicking *Re-open*.

4.5.3 Summary Table

On the reviews page below the description is a table which lists all of the issues found in reviews.



Issue Summary:

Total: 9 Open: 1 Resolved: 7 Dropped: 1

Status: All **From:** All

Issue ID	Description	From	Last Updated	Status
Diff 7699	This would be better as an underscore template cached outside the function, so it's compiled only once. When doing lots of string concatenations in JS, it's best to do: ['...', '...'].join('')	Christian Hammond	Nov. 30, 2012, 1:54 a.m.	Resolved
Diff 7700	Should be able to do .children('br'), I think.	Christian Hammond	Nov. 30, 2012, 1:54 a.m.	Resolved
Diff 7701	Like above, if we can have a static variable that we set only once (probably after first call), this will be nicer. And same comment about concatenation.	Christian Hammond	Nov. 30, 2012, 2 a.m.	Resolved
Diff 7702	Let's just test truthiness, in case this ends up normalized to null or something.	Christian Hammond	Nov. 30, 2012, 1:47 a.m.	Dropped
Diff 7704	I feel like this all belongs in the template. You can do: <% if blah { %> ... <% } %> The nice thing about that is, when we move to require.js, templates like these can be stored in their own template files and we have a nice separation of templates and logic.	Christian Hammond	Nov. 30, 2012, 1:54 a.m.	Resolved
Diff 7703	Same.	Christian Hammond	Nov. 30, 2012, 1:54 a.m.	Resolved
Diff 7765	Needs to be a var.	Christian Hammond	Dec. 9, 2012, 2:28 a.m.	Open
Diff 7766	var	Christian Hammond	Dec. 1, 2012, 8:54 p.m.	Resolved
Diff 7767	Should be: /* * */	Christian Hammond	Dec. 1, 2012, 8:56 p.m.	Resolved

Clicking the *Status* drop-down will allow you to choose between *Open*, *Dropped*, *Resolved*, or *All issues*. Clicking the *From* drop-down will allow you to filter the list by individual reviewers. Clicking on a row will jump to the comment.

4.6 Review Draft Banner

Any time you have an unfinished draft of a review, you will see a green draft review banner docked to the top of your browser window on any pages associated with that review request. This will help you to remember that you still have a review pending, and lets you quickly interact with it.

The banner has the following buttons:

- *Edit Review*
- *Publish Review*
- *Discard*

Clicking *Edit Review* will display the *review dialog* (page 29), allowing you to make changes to your review.

Clicking *Publish Review* will immediately *publish your review* (page 29).

Clicking *Discard Review* will immediately discard your review.

4.7 Creating and Editing Reviews

4.7.1 Creating Reviews

Adding comments on diffs or file attachments will immediately create a new review if one doesn't already exist. Once a review is created, a *review draft banner* (page 28) will show on the page.

Blank reviews can also be created without having to first add a comment. This can be useful if you have a generic comment about the change, rather than one specific to diffs or other files.

To create a new, blank review, click *Review* on the review request actions bar.

4.7.2 Editing Reviews

Clicking the *Review* button on the review request action bar, or clicking *Edit Review* on the review draft banner will display dialog showing your review.

In this dialog, you can add a summary for your review, *approve it* (page 29) for shipping, and edit your existing comments.

The bottom of the dialog have the following buttons:

- *Publish Review*
- *Discard Review*
- *Cancel*
- *Save*

Clicking *Publish Review* will save any comments you have made on your review and immediately publish your review.

Clicking *Discard Review* will immediately discard your review.

Clicking *Cancel* will discard any new changes you've made in the dialog, but keep your review around if this wasn't a new one.

Clicking *Save* will save any comments you have made on your review and close the dialog.

4.8 Publishing Reviews

You can publish your review by clicking *Publish Review* on the *review draft banner* (page 28) or *review dialog* (page 29).

Publishing your review will add a review to the review request page showing all your comments. The developer or other reviewers can reply to your comments, discussing the changes with you.

If the administrator has enabled e-mail support, the owner of the review request and all existing reviewers will be e-mailed a copy of your new review.

Once you have published a review, you can't go back and edit it.

4.9 Approving Changes (Ship It!)

To mark your approval for a change to be submitted on your review, check the *Ship It* checkbox on the review dialog. Your review will have a green "Ship It!" label indicating your go-ahead for the developer to submit the change.

It is up to your organization to decide how Ship Its are used. Review Board doesn't enforce any policy about who can mark a review as Ship It, and doesn't require a Ship It to close a review request.

Once a review has been marked Ship It and published, it cannot be undone.

4.9.1 Quick Ship-It

New in version 1.6.

If you're approving a change and you don't have any other comments to add, you can click the *Ship It!* button on the review request action bar instead. A confirmation dialog will appear asking if you're sure, and once given permission, it will create and publish a new review saying "Ship It!"

4.10 Replying to Comments

All reviews (including comments on code and file attachments) are shown on the review request page in individual review boxes. Discussion about a particular review will take place solely within that review box.

To reply to a comment in a review, click the *Add comment* link below the comment on the right-hand side of the box. You can type your comment and then press *OK* to save, or *Cancel* to discard the comment.

You can use rich text in your replies using the *Markdown* language. See *Using Markdown* (page 35) for more information.

Unpublished comments (as indicated by your name in green) will appear only to you. To publish your replies, click *Publish* in the reply draft banner at the top of that review box.

Depending on the server setup, replying to a review may send out an e-mail to all people involved in that review. This is displayed as part of a threaded discussion in the e-mail clients. Because of this, you must publish replies on a per-review basis. It's like replying to separate e-mail threads in a mail client.

CHAPTER

FIVE

SEARCHING

5.1 Quick Search

New in version 1.6.

The search field found on every page can be used to quickly locate review requests, users, and groups without needing to do a full search.

As you type, a list of matches will appear. Clicking on a match will immediately take you to the appropriate page.

If your server has *full-text search* (page 31) enabled, hitting enter after typing your search terms will perform a full search.

The following can be searched using Quick Search:

- Review request IDs
- Review request summaries
- Usernames
- Users' first/last names
- Group names
- Group display names

5.2 Full-Text Search

If your Review Board server has full-text search enabled, you will be able to search through all public review requests using the search field on any page. This will match text in the review request, the files modified, and other fields using our query syntax.

Logical operators, such as AND, OR, and NOT, can be used to narrow down your search. Individual fields on review requests can also be searched.

5.2.1 Query Syntax

There are a variety of ways to combine terms in the search field. By default, the search results will be an "AND" of any words entered in the box. This means searching for `window javascript` will give pages that have both of those terms in them.

In order to narrow down your results, there are a few useful operators you can use.

- **OR**:

  This operator will change the relationship from "AND" to "OR". This will make it so search results will contain any of the words instead of all. Searching for `window OR javascript` will yield all review requests that contain either of those words.

- **NOT**:

  This works a lot like `OR`, except it will filter out results containing the NOT term. For example, `window NOT javascript` will return matches that have "window" but not "javascript".

- **Phrase**:

  Sticking something in double-quotes will search for the exact phrase instead of splitting it up into terms.

There are a number of other operators you can use to tweak the results. For a full explanation of the Whoosh query syntax (including a couple features not mentioned here), see The default query language (http://pythonhosted.org/Whoosh/querylang.html).

5.2.2 Fields

In addition to searching the full text of a review request, you can search individual fields for better results. To search for a term inside a specific field, prefix that term with *field*:, where *field* is one of the below:

- `summary`:

  This field searches only the summary. `summary: window` will match requests with window in the summary only.

- `description`:

  This field searches only the description. `description: javascript` will match requests with javascript in the description only.

- `testing_done`:

  This field searches only Testing Done. `testing_done: tested` will match requests with tested in Testing Done only

- `author` and `username`:

  These two fields search the review request poster. `author` will search both the username and full name, whereas `username` is just the username.

- `bug`:

  This field searches by bug identifier. Searching for `bug:83724` will find any review requests which address that bug.

- `file`:

  This field indexes filenames in the diff. Searching for `file:frob.c` will yield any review requests which altered that file.

These fields can be combined like any other terms. Searches like `file:frob.c AND author:Jim` can make it easy to quickly find old review requests.

5.2.3 What's Indexed

The full text of review requests, including the summary, description, and testing, is indexed.

Lists of files in the diffs are also indexed. The contents of the diffs are not.

Reviews are not indexed.

Private review requests are not indexed.

CHAPTER

SIX

USING MARKDOWN

New in version 2.0.

Many of the multi-line text fields in Review Board support a simple markup language called *Markdown*. This allows you to perform basic formatting of your text (such as creating lists or denoting emphasis), as well as more complex things like including syntax-highlighted code samples or images.

This document does not intend to be a full reference on the Markdown language, but rather a quick primer on the basic features that are useful when writing review requests or reviews.

Note: Review Board's implementation of Markdown shares a lot in common with GitHub Flavored Markdown (https://help.github.com/articles/github-flavored-markdown). While it's part of the basic Markdown spec, embedding raw HTML is *not* allowed, to prevent cross-site scripting attacks. If you include HTML tags, they will be shown to the user as-is, rather than treated as HTML.

6.1 Basic Markdown Syntax

6.1.1 Headers

Headers are added by underlining the relevant text with equals signs or dashes:

```
Header
======
```

6.1.2 Lists

Markdown supports both ordered (numbered) and unordered (bulleted) lists. These are written using a natural syntax. Ordered lists use numbers followed by periods:

```
1. First item
2. Second item
3. Third item
```

While unordered lists can be defined with asterisks, plus signs, or hyphens:

```
* First item
* Second item
* Third item
```

6.1.3 Emphasis

Text can be emphasized by surrounding it with asterisks or underscores. The resulting text will be shown in a heavier font:

```
This text is *italic*

This text is **emphasized**
```

6.1.4 Links

Basic links can be added to your text using a combination of square brackets and parentheses:

```
This text has a [link to wikipedia](http://wikipedia.org/).
```

Note: In most cases, you won't need to build your links yourself. Any URLs that are included in your text will automatically be turned into links. In addition, certain special strings like "bug 234" or "/r/583" will be automatically linked to the relevant bug or review request.

6.1.5 Images

If you have images which are accessible from a URL, you can embed them into your text using a syntax similar to links. These start with an exclamation mark, followed by square brackets containing the "alt" attribute, followed by parentheses with the URL to the image:

```
![Image description](http://example.com/image.png)
```

6.1.6 Tables

Simple tables can be inserted by drawing the table using a combination of vertical bars and hyphens:

```
Header | Header | Header
-------|--------|-------
Cell   | Cell   | Cell
Cell   | Cell   | Cell
```

6.2 Code Samples

When writing reviews, It's often very useful to write small snippets of code. Markdown allows you to notate which parts of your text are code, and the result will be rendered with syntax highlighting. This can be especially nice for proposing changes.

Code can be formatted inside a line by enclosing the text in single backticks. This is often useful when referring to symbols from the code:

```
I think it would be nice if you moved this code up near the `do_foo` method.
```

Longer code samples can be denoted using block notation. Any blocks which are indented at least 4 spaces will be treated as a code block. In addition, code blocks can be notated without indentation by surrounding the block with triple backticks, using the syntax from GitHub Flavored Markdown. This also allows you to explicitly state what language the code is written in, for syntax highlighting:

```
Here's an indented code block:

    def test():
        logging.error('Test failed')

And a "fenced" code block:

```javascript
function test() {
    console.log('Test failed');
}
```
```

6.3 Escaping

Because Markdown syntax endows many common punctuation symbols with special meaning, these can sometimes unintentionally trigger formatting. In this case, you can avoid this by escaping the relevant character with a backslash:

```
I really want a \` backtick in this line.
```

Backslash escapes can be used for the following characters:

```
\ backslash
` backtick
* asterisk
_ underscore
{} curly braces
[] square brackets
() parentheses
# hash mark
+ plus sign
- minus sign
. period
! exclamation mark
```